

## EE432: Digital Signal Processing Fall 2012

### **Project 01: MATLAB Refresher**

**Assigned: Thurs 8/23/2012**

**Due: Thurs 8/30/2012**

#### **Introduction**

In this project, you will refresh your memory with the functions that are available in MATLAB.

#### **I. Elementary Functions**

Using MATLAB, find the values of:

1.  $\csc(95^\circ)$  \_\_\_\_\_
2.  $e^{-2.4}$  \_\_\_\_\_
3.  $|3 + j4|$  (magnitude) \_\_\_\_\_
4.  $\tan^{-1}(\infty)$  \_\_\_\_\_ (put your answer in degrees)
5.  $\sqrt[4]{101}$  (fourth root) \_\_\_\_\_
6.  $\sum_{k=-499}^{500} k^3$  (sum of cubes) \_\_\_\_\_
7.  $11!$  (factorial) \_\_\_\_\_
8.  $\ln(150)$  (natural log) \_\_\_\_\_
9.  $0 \oplus 0 =$  \_\_\_\_\_       $0 \oplus 1 =$  \_\_\_\_\_ (Note: XOR)  
 $1 \oplus 0 =$  \_\_\_\_\_       $1 \oplus 1 =$  \_\_\_\_\_
10.  $\lceil 5.5 \rceil =$  \_\_\_\_\_ (ceiling)       $\lfloor -5.5 \rfloor =$  \_\_\_\_\_ (floor)  
 $\text{ROUND}(14.4) =$  \_\_\_\_\_       $\text{mod}(655,51) =$  \_\_\_\_\_ (modulo 51)

#### **II. Dual-tone Multi-Frequency Signals (DTMF)**

The MATLAB *phone* function demonstrates the sounds used by touch tone phones. When you enter a phone number to place a call, each time you press a number button you are essentially transmitting 50msec of the sum of a pair of sinusoids to the phone company, letting them know the phone number you wish to connect to. As long as the phone company can identify the two tones, it can determine the digits of the desired phone number. Fourier analysis can be used to detect the two tones that are present so as to identify each digit. This use of Fourier analysis will be used later in

the course.

Start the MATLAB *phone* function by typing `>> phone` at the command line. Enable the sound so you can hear the different tones by selecting the “Sound” check box. Punch in a few numbers and listen to the tones. At the same time, watch the time signal plot and frequency spectrum plot and see if you can pick out the two tones (sinusoidal frequencies) that are present with each digit. A table that summarizes the tones associated with each button on a touch-tone phone is shown below. Note that the last column is not used on a typical touch-tone phone (digits A, B, C and D), but can be used for data transmission.

		High-Group Frequencies			
		1209Hz	1336Hz	1477Hz	1633Hz
Low-Group Frequencies	697Hz	1	ABC 2	DEF 3	A
	770Hz	GHI 4	JKL 5	MNO 6	B
	852Hz	PRS 7	TUV 8	WXY 9	C
	941Hz	*	OPER 0	#	D

Table of sinusoidal frequencies for a DTMF system.

Important: the equation for a sinusoid (sine or cosine) that has a frequency of  $f_0$  Hz is written as

$$S=A \cos(2\pi f_0 t),$$

where  $A$  is the amplitude and  $t$  is the time variable. This means that if you want to create a cosine wave of 100 Hz, with amplitude 2, you should create a time vector and write the cosine as:

$$s=2*\cos(2*pi*100*t);$$

Since DTMF tones are pairs of sinusoids, for each digit of a phone number you would need to create two separate sinusoids (use cosines), with the two frequencies as given in the table above, and add them together. We must be concerned that the sinusoids we create are subject to the *Nyquist* A/D sampling criteria. The minimum sample rate to be used with DTMF tones, given the table above is  $2 \times 1633 \text{ Hz} = 3266 \text{ samples/sec}$  (or also referred to as 3266 Hz). This is because 1633 Hz is the highest-frequency sinusoid we will use. For our work, we will actually use a sample frequency higher than the Nyquist rate,  $f_s = 8000 \text{ Hz}$ .

1. Create a signal called  $s7$  which is the DTMF tone for the digit ‘7’ that lasts for 0.5 seconds. Note that in order to create the DTMF tone, you must create a time vector that runs from 0 to 0.5 sec, with a sample spacing of  $1/f_s$  seconds.

2. Listen to this tone using the MATLAB *soundsc* function. If  $s7$  is the DTMF signal you created, then the following command will play it so you can hear it through the speakers:

```
soundsc(s7,8000);
```

Note: you must pass *soundsc* the correct sample frequency of 8000.

3. Now create a signal  $s3$  which is the DTMF tone for the digit ‘3’ that lasts for 250 msec. You would need a new time vector that lasts only from 0 to 250 msec. Listen to the tone.

4. Create a signal  $z$  that represents a period of 250 msec of silence (zero value) using the same sample rate. Note that the `zeros` command can create a vector or matrix of zero values. You can try to listen to it also, but you should hear nothing.
5. Now join  $s3$ ,  $z$ , and  $s7$  together into a new signal and which represents  $s3$  followed by silence ( $z$ ) followed by  $s7$ , for example  $S = [s3 \ z \ s7]$  or  $S = [s3, z, s7]$ . Listen to the new signal.
6. Now add a third DTMF signal, 0.35 seconds of the DTMF digit '5'. Note that you will need a new time vector. Play all three DTMF tones as one signal (ensure you have a period of silence in between).

### III. A Function Library

1. Write a MATLAB function called `IsOdd` that will determine if an input integer is odd or not. You would use it as in:

```
>> y = IsOdd(x);
```

where  $x$  is the integer input value, and  $y$  is equal to 1 if  $x$  is odd, and 0 otherwise. If  $x$  is not an integer, this function should return  $y = []$  (which is the NULL value) and display a warning message.

Helpful considerations:

How do you tell if a number is an integer?

How do you tell if an integer is odd?

2. Write a MATLAB function called `iseven` that will determine if an input integer is even or not. You would use it as in:

```
>> y = IsEven(x);
```

where  $x$  is the integer input value, and  $y$  is equal to 1 if  $x$  is even, and 0 otherwise. If  $x$  is not an integer, this function should return  $y = []$  (which is the NULL value) and display a warning message. There are several approaches that would work. USE YOUR ISODD FUNCTION INSIDE `IsEven.m`.

3. Write a MATLAB function called `CreateDtmf` that will create a signal that is a specified DTMF tone, lasting for a specified time period, sampled at 8000 Hz. You would use it as in:

```
>> y = CreateDtmf(x, td);
```

where  $x$  is the desired DTMF signal,  $td$  is the time duration in seconds, and  $y$  is the output DTMF tone. If  $x$  is not one of the DTMF signals specified below, or if  $td$  is not  $> 0$ , this function should return  $y = []$  (which is the NULL value) and display a warning message. The allowable desired DTMF signals are:

```
'1', '2', '3', 'A'
```

```
'4', '5', '6', 'B'
```

```
'7', '8', '9', 'C'
```

```
'*', '0', '#', 'D'
```

Important: This function **SHOULD NOT** have a `soundsc` command in it. The only purpose of this function is to create a signal that is a proper DTMF signal and lasts for the specified duration. In order to hear the DTMF tone, you should listen to it **OUTSIDE** of the function.

4. Write a MATLAB function called *CreateSilence* that will create a signal that is all zeros for a specified time period, sampled at 8000 Hz. You would use it as in:

```
>> y = CreateSilence(td)
```

where *td* is the time duration in seconds, and *y* is the silence signal. If *td* is not  $> 0$ , this function should return *y* = [ ] (which is the NULL value) and display an error message.

Important: This function **SHOULD NOT** have a *soundsc* command in it. The only purpose of this function is to create a signal that is a silence signal and lasts for the specified duration.

#### IV. Test Your Function Library

1. Using your both your even/odd functions, make sure your functions work by recording your answers to these problems:

Input value	<i>IsOdd</i> output:	<i>IsEven</i> output:
4		
-5		
1.3		
pi		

2. Write a program that will create and play all 16 of the DTMF tones for 100 msec each, with 50 msec of silence in between each tone. Demonstrate that your program plays these tones for the professor, and get the professor to initial here: \_\_\_\_\_

**For this project's write-up:**

- Turn in your answers to part I (hard copy)
- Play your DTMF tones for part II
- Turn in your code for part III (hard copy)
- Turn in your answers to the table in part IV (hard copy)
- Play your DTMF tones for part IV